Senai Cimatec

Sam de Almeida

Alta Qualidade de Software - Prática Obrigatória II - Relatório de Bugs

Salvador, Bahia
2025

Sumário

Lista de Figuras

1 Soma incorreta devido a duplicação da adição

1.1 Localização

O bug está no arquivo calculadora.py na função adicao.

1.2 Descrição do bug

O bug faz com que a função retorne a soma de x pela soma de x + y, duplicando a soma e entregando o valor incorreto ao usuário.

1.3 Contexto

Ter instalado Python 3.x e NumPy 1.26.x.

1.4 Passo a passo

- Executar o programa
- Digite 1 para operação
- Digite 5
- Digite 8
- O resultado será 18

1.5 Consistência

O erro ocorrerá sempre.

1.6 Resultado esperado vs obtido

Considerando os valores 5 para o primeiro número e 8 para o segundo, o resultado deveria ser 13, porém retorna o valor 18.

1.7 Evidência

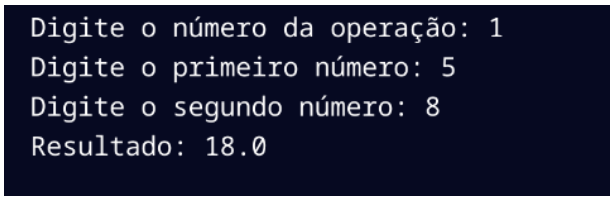

Figura 1: Evidência do erro na função adicao

1.8 Logs

Não há logs gerados.

1.9 Severidade

A severidade do bug é média.

1.10 Impacto

Impede que o usuário consiga fazer somas.

1.11 Possíveis soluções

Uma solução possível é retirar a primeira parte, deixando apenas a função np.add(), assim tendo apenas a função de adição da biblioteca numpy. Outra solução é tirar a função np.add(x + y) e substituir somente por y, assim garantindo outra soma simples.

1.12 Data de identificação

27/05/2025.

1.13 Status

Não resolvido.

2 Falta de tratamento para divisão por 0

2.1 Localização

Função divisao no arquivo calculadora.py.

2.2 Descrição do bug

A função não trata a divisão por zero, resultando em ZeroDivisionError e encerramento do programa.

2.3 Contexto

Ter instalado Python 3.x.

2.4 Passo a passo

- Execute o programa
- Digite 4 para operação
- Digite qualquer valor para o primeiro número
- Digite zero para o segundo número
- O programa irá encerrar e apresentará o erro ZeroDivisionError

2.5 Consistência

O erro ocorre sempre que o segundo valor inserido for 0.

2.6 Resultado esperado vs obtido

O resultado esperado é um aviso de que não é possível dividir números por 0. O resultado obtido é um erro e a interrupção do programa.

2.7 Evidências

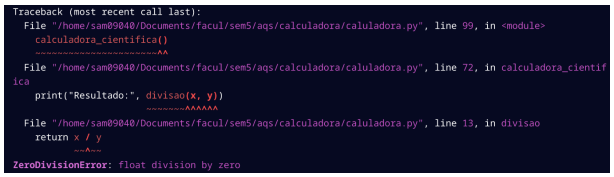

Figura 2: Evidência do erro na função divisao

2.8 Logs

Traceback (most recent call last):
File ”calculadora/calculadora.py”, line 99, in <module>
calculadora_cientifica()
File ”calculadora/calculadora.py”, line 72, in calculadora_cientifica
print(”Resultado:”, divisao(x, y))
File ”calculadora/calculadora.py”, line 13, in divisao
return x / y
ZeroDivisionError: float division by zero

2.9 Severidade

Crítica.

2.10 Impacto

Encerra a execução do código, impedindo que o usuário continue a utilizá-lo.

2.11 Possíveis soluções

Implementar uma verificação se o y tem valor igual a 0 na função divisao.

2.12 Data de identificação

27/05/2025.

2.13 Status

Não resolvido.

3 Lógica incorreta no cálculo do fatorial

3.1 Localização

Função fatorial no arquivo calculadora.py

3.2 Descrição do bug

A função inclui o zero na multiplicação inicial e inicia o loop em 0, o que faz com que o resultado final sempre seja 0.

3.3 Contexto

Ter instalado Python 3.x.

3.4 Passo a passo

- Execute o programa
- Digite 7 para a operação
- Digite qualquer valor
- O resultado será 0

3.5 Consistência

Ocorrerá com qualquer valor acima de 0 dado ao programa.

3.6 Resultado esperado vs obtido

Tomando 3 como valor dado, o resultado deveria ser 6, porém retorna 0.

3.7 Evidências

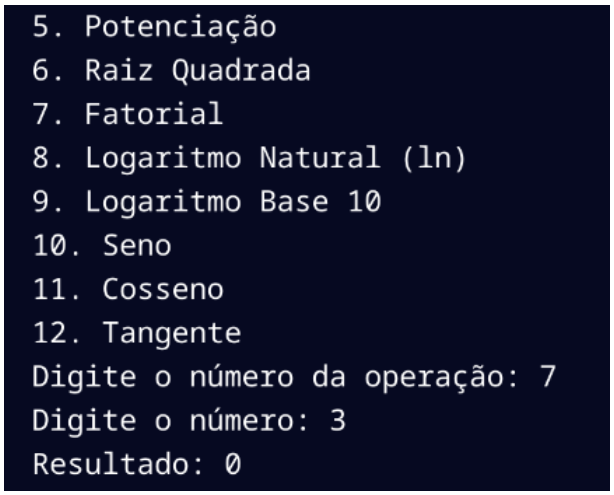

Figura 3: Evidência do erro na função fatorial

3.8 Logs

Não gera logs.

3.9 Severidade

Alta.

3.10 Impacto

Gera resultado incorreto para todos os casos em que o valor for maior que 0.

3.11 Possíveis soluções

Uma possível solução é alterar a função, alterando o for para começar por 1.

3.12 Data de identificação

27/05/2025.

3.13 Status

Em análise.

4 Função de logaritmo natural utilizando a função errada do numpy

4.1 Localização

Função logaritmo_natural no arquivo calculadora.py.

4.2 Descrição do bug

Ao tentar utilizar a função de logaritmo natural, utiliza np.ln(), função essa que não existe no Numpy, gerando o encerramento do programa.

4.3 Contexto

Ter instalado Python 3.x e Numpy 1.26.x.

4.4 Passo a passo

- Execute o programa
- Digite 8 para operação
- Digite qualquer valor
- O programa encerrará e retornará o erro AttributeError

Rode o programa, digite 8 na escolha de operação e digite qualquer valor, assim o programa encerrará e retornará a mensagem: AttributeError: module 'numpy' has no attribute 'ln'.

4.5 Consistência

Sempre ocorre.

4.6 Resultado esperado vs obtido

Considerando o valor a ser calculado como 1, o resultado deveria ser 0, porém retorna o erro AttributeError.

4.7 Evidências

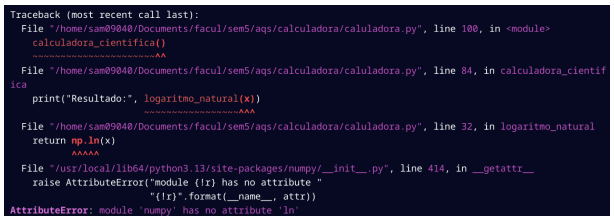

Figura 4: Evidência de erro na função logaritmo natural

4.8 Logs

Traceback (most recent call last):
File ”calculadora/calculadora.py”, line 100, in <module>
calculadora_cientifica()
File ”calculadora/calculadora.py”, line 84, in calculadora_cientifica
print(”Resultado:”, logaritmo_natural(x))
File ”calculadora/calculadora.py”, line 32, in logaritmo_natural
return np.ln(x)
File ”/usr/local/lib64/python3.13/site-packages/numpy/___init___.py”,
line 414, in ___getattr___
raise AttributeError(”module !r has no attribute ”
”!r”.format(___name___, attr))
AttributeError: module ’numpy’ has no attribute ’ln’

4.9 Severidade

Crítica.

4.10 Impacto

Encerra o código; função completamente inutilizável.

4.11 Possíveis soluções

Uma opção é trocar o nome da função de ln para log, o nome correto para utilizar a função de logaritmos naturais da biblioteca numpy.

4.12 Data de identificação

27/05/2025.

4.13 Status

Não resolvido.

5 Função retorna logaritmo natural em vez de base 10

5.1 Localização

Na função logaritmo_base10 no arquivo calculadora.py.

5.2 Descrição do bug

Quando o usuário escolhe a opção de logaritmo base 10 e insere um valor, a função retorna o logaritmo natural do valor, não o base 10.

5.3 Contexto

Ter instalado Python 3.x e Numpy 1.26.x.

5.4 Passo a passo

- Execute o programa
- Digite 9 para operação
- Digite o valor 10
- O programa retornará o valor 2,302585093

5.5 Consistência

Sempre ocorre.

5.6 Resultado esperado vs obtido

Ao utilizar o valor 10 como referência e executar o programa, espera-se que o resultado da conversão pelo logaritmo de base 10 seja 1. No entanto, o programa retorna 2,302585093.

5.7 Evidências

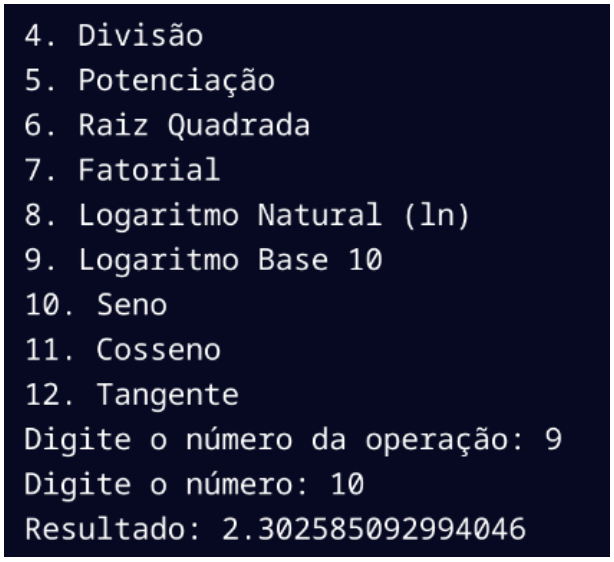

Figura 5: Evidência de erro na função logaritmo de base 10

5.8 Logs

Nenhum log gerado.

5.9 Severidade

Média.

5.10 Impacto

Gera resultados incorretos.

5.11 Possíveis soluções

Alterar a função para retornar np.log10().

5.12 Data de identificação

27/05/2025.

5.13 Status

Corrigido.

5.14 Código corrigido

Listing 1: Código corrigido da função logaritmo_base10

```
def logaritmo_base10(x):
if x <= 0:
    return "Erro:␣Logaritmo␣de␣número␣não␣positivo"
return np.log10(x)
```

Figura 6: Teste pós-correção

6 Chamadas para funções inexistentes: seno, cosseno e tangente

6.1 Localização

Arquivo calculadora.py nos blocos elif escolha == ’10’, ’11’ e ’12’ na função calculadora_cientifica.

6.2 Descrição do bug

As funções trigonométricas são chamadas no código, mas não foram definidas em lugar nenhum, resultando em erro de execução.

6.3 Contexto

Ter instalado o Python 3.x.

6.4 Passo a passo

- 1ª opção:
  - Execute o programa
  - Digite 10 para operação
  - O programa encerrará sem nenhum retorno
- 2ª opção:
  - Execute o programa
  - Digite 11 ou 12 para operação
  - Digite qualquer valor
  - O programa encerrará e retornará o erro NameError.

6.5 Consistência

Sempre ocorre.

6.6 Resultado esperado vs obtido

O resultado esperado é o cálculo correto da opção escolhida. O resultado obtido é o erro NameError.

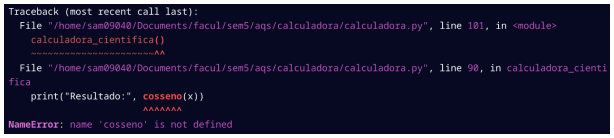

Figura 7: Evidência de erro causado por falta das funções de trigonometria

6.7 Evidências

6.8 Logs

Traceback (most recent call last):
File ”calculadora/calculadora.py”, line 101, in <module>
calculadora_cientifica()
File ”calculadora/calculadora.py”, line 90, in calculadora_cientifica
print(”Resultado:”, cosseno(x))
NameError: name ’cosseno’ is not defined

6.9 Severidade

Crítica.

6.10 Impacto

Interrupção completa da execução.

6.11 Possíveis soluções

Implementação das funções seno, cosseno e tangente, assim como adicionar a opção da função seno no menu.

6.12 Data de identificação

28/05/2025

6.13 Status

Não resolvido.

7 Correção utilizando TDD: função logaritmo_base10

7.1 Correção aplicada

Substituído np.log() por np.log10().

7.2 Metodologia

- Aplicado TDD
  - Revisão do bug: O relatório do bug mostrava que a função logaritmo_base10() utilizava np.log(), retornando o log natural
  - Entendimento do problema: O erro causava o retorno incorreto para entradas como 100, cuja saída deveria ser 2, porém retornava 4,605170186
  - Configuração do ambiente: Usado Python 3.x, Numpy 1.26.x e Unittest para testes automatizados
  - Escrita do teste: Foi criado o seguinte teste com unittest:

    Listing 2: Código testando a função logaritmo_base10

    ```
    class TestCalculadora(unittest.TestCase):
    def teste_logaritmo_base10(self):
    # Teste com valor 100
    resultado = logaritmo_base10(100)
    esperado = 2.0
    self.assertEqual(resultado, esperado,
    f"Esperado:{esperado},␣mas␣obtido:{resultado}")
    ```

  - Execução do teste: O teste falhou inicialmente, confirmando o bug
  - Correção mínima: Substituído np.log() por np.log10() na função.
  - Reteste: O teste passou após a correção
- Refatoração: Mensagem de erro ajustada para incluir x <= 0 e código limpo;
- Validação final: Todos os testes passaram após a refatoração;

7.3 Ferramentas

Unittest e Numpy.

7.4 Status

Corrigido e validado com testes.

Figura 8: Código final corrigido